

Towards Scalable Autonomous Middleware Infrastructure

Dorian Arnold
Computer Science Department
University of New Mexico

Life Before ICL



MY ICL Tenure

- ▶ January 1999 – August, 2001
- ▶ NetSolve Project
- ▶ ICL Collaborators:
 - Henri, Sudesh, Sathish, Jakob, Thara, Michelle, Dieter, Keith S., Tsinghua, Victor, Susan, Shirley, Keith M., Ganapathy, Nathan, David

Life After ICL

- ▶ PhD Student, U. of Wisconsin, '01 - '08
 - Scalable, reliable communication infrastructure
 - Scalable, lightweight tools and applications
- ▶ Asst. Professor, U. of New Mexico, since '09
 - Carry-over from dissertation work (naturally)
 - Autonomous Infrastructure
 - Virtualization for HPC
 - Thin Computing

The Method behind the Madness

- ▶ What is HPC?
 - It depends ... on who you ask?
 - Computer architect?
 - OS researcher?
 - Applied mathematician? (Dead-end career 😊)
 - Domain scientist?
 - **Distributed systems researcher!**
- ▶ Many domains of expertise
 - One should not need cross-domain expertise to use HPC resources effectively

My Research Foci

Make HPC systems easier to use without sacrificing performance and reliability

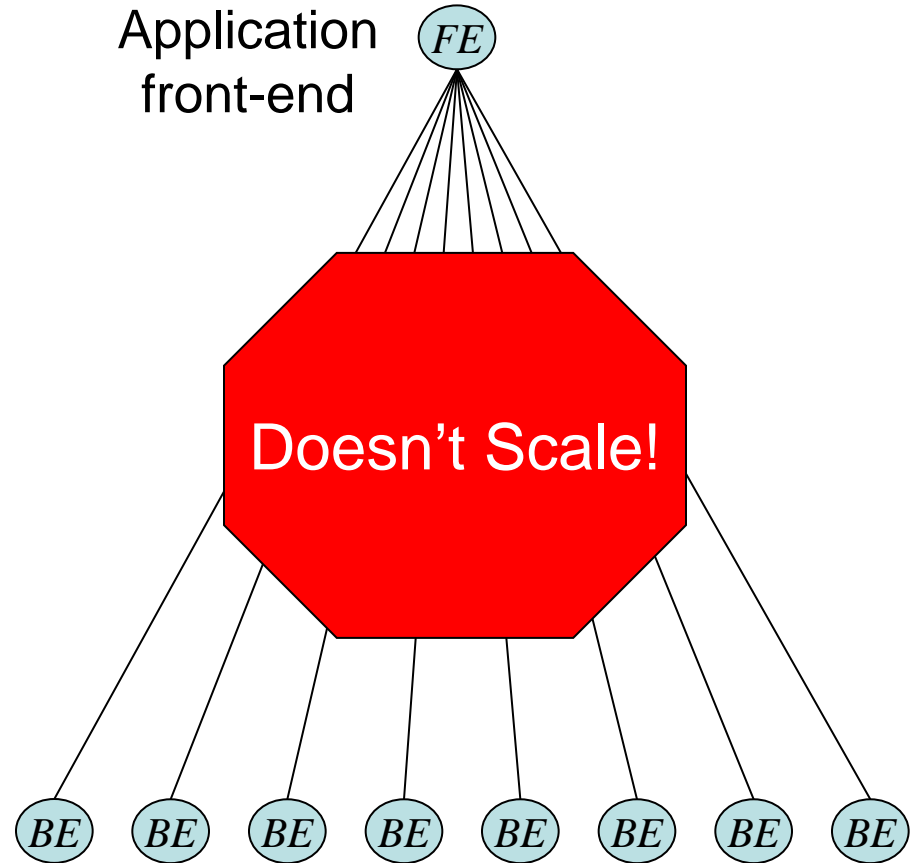
Scalable Communication Infrastructure

Tree-based Overlay Networks

We need Scalable Tools and Applications

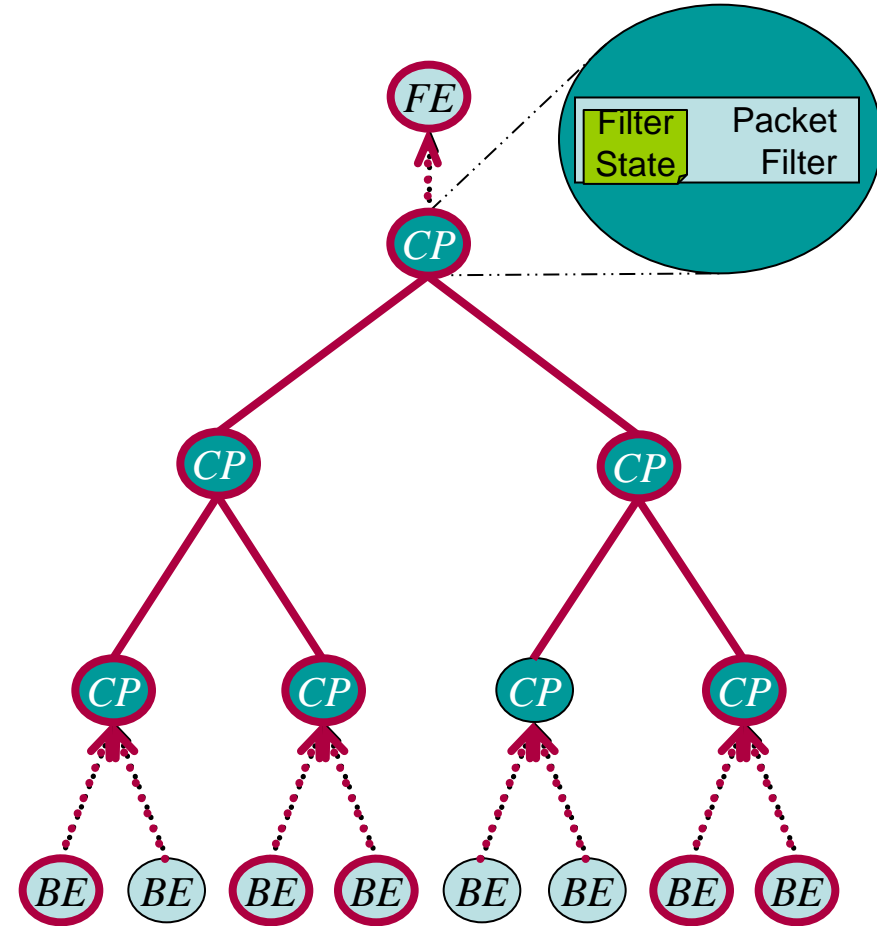
- Data management
 - Large volumes
- Data analysis
 - Centralized analysis leads to computational bottlenecks
- Many resources to manage
 - E.g. control channels

Application
back-ends



Key TBON Abstractions

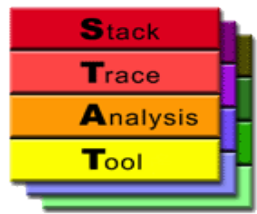
- ▶ Filters
 - Executed by processes
 - Persistent state
- ▶ Channels
 - Reliable
 - Order-preserving
- ▶ Streams
 - Define sub-groups
 - Distinguish dataflows
 - Specify filter routine



The Multicast/Reduction Network

- ▶ MRNet is our prototype TBON
 - Developed by Arnold, Roth and Miller
- ▶ Used by many research installations
 - Paradyn (University of Wisconsin)
 - Stack Trace Analysis Tool (LLNL)
 - TauOverMRNet (University of Oregon)
 - TBON-FS (University of Wisconsin)
 - Image Analysis (University of Wisconsin)
 - CEPBA-Tools (Universitat Politècnica de Catalunya)
 - Open|SpeedShop (Krell Institute)
 - TotalView (TotalView Tech.)
- ▶ Ongoing collaborations: RENCI, Juelich
 - Scalable Tool Communication Infrastructure???????

Stack Trace Analysis Tool



Stack Trace Analysis Tool

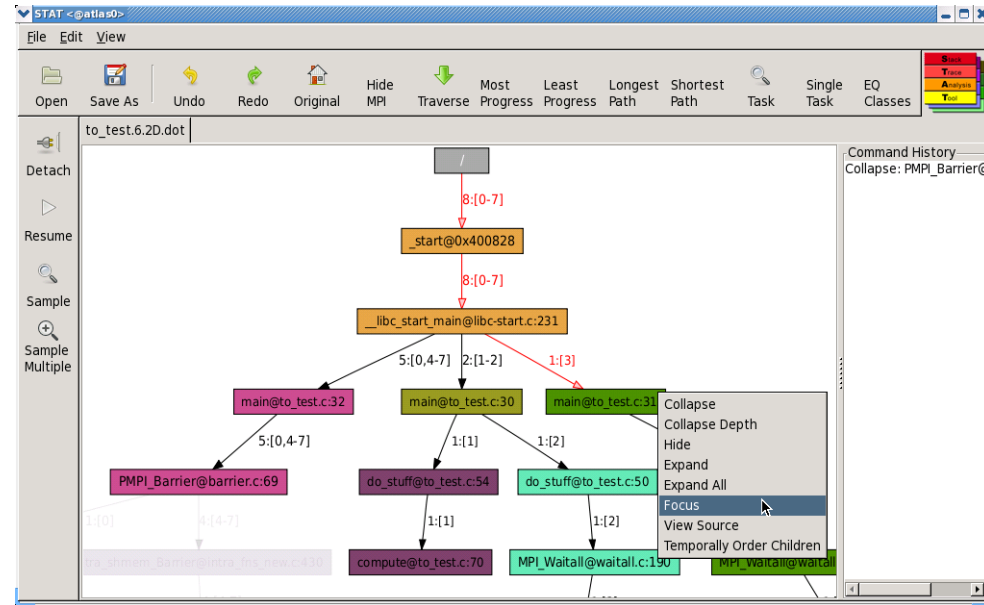
Extreme Scale Debugging

STAT is a **lightweight** debugging aid that uses **stack traces** to **classify** process equivalence and profile application.

Thousands of tasks **reduce** to few classes.

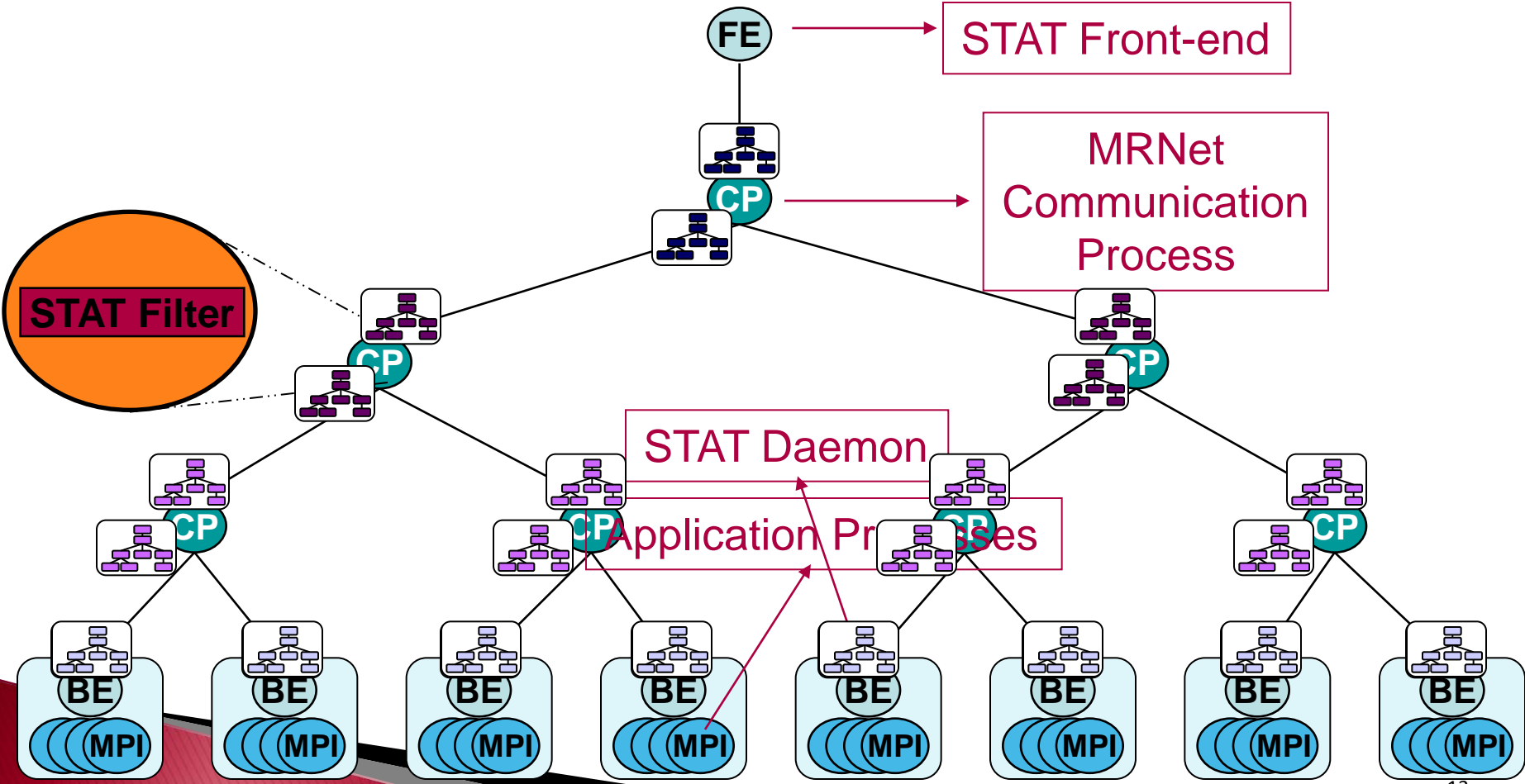
Analyze **representatives** with full debugger

Temporal analysis determines tasks' relative progress

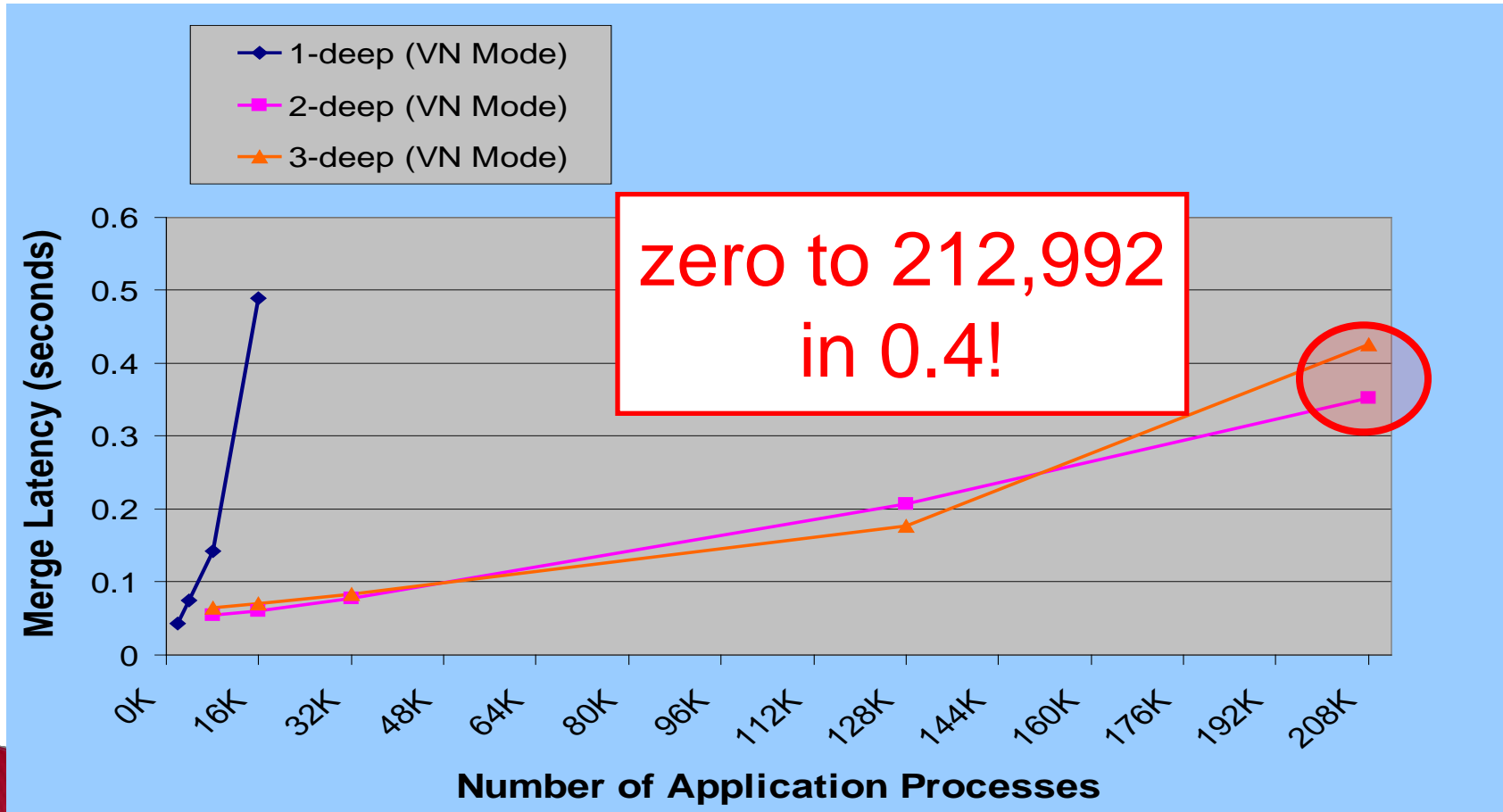


Goal: Scale **up** to machine sizes and scale **down** the information

Stack Trace Analysis Tool

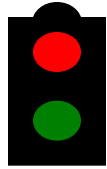


STAT Performance on BlueGene/L

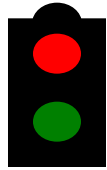


Scalable, Robust Data Aggregation

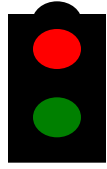
Observations and Recovery Approach



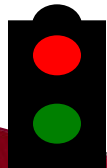
Explicit state replication is expensive
Use inherent information redundancy



Strong data consistency is expensive
Use weak data consistency



Global coordination is expensive
Use localized protocols to satisfy global requirements

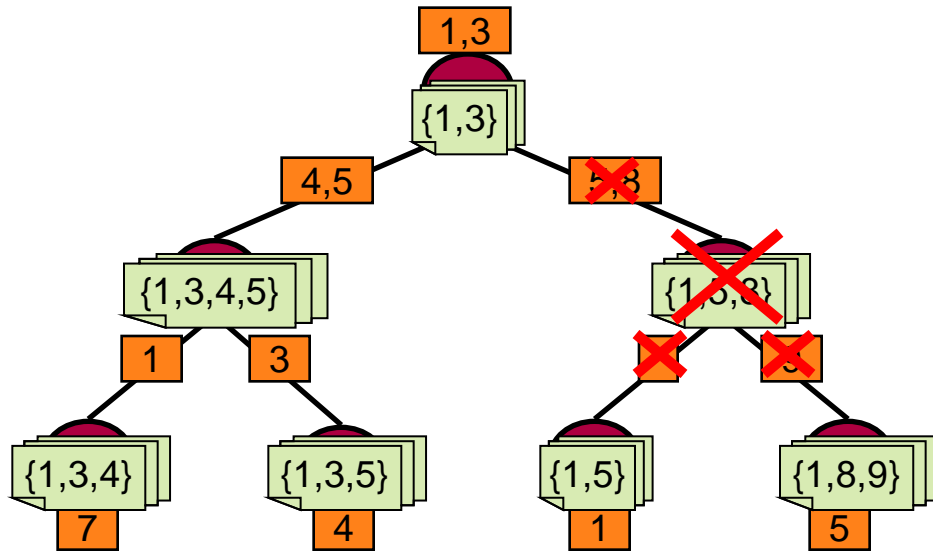


Information must be disseminated globally
Use TB• N for efficient, scalable dissemination

State Compensation

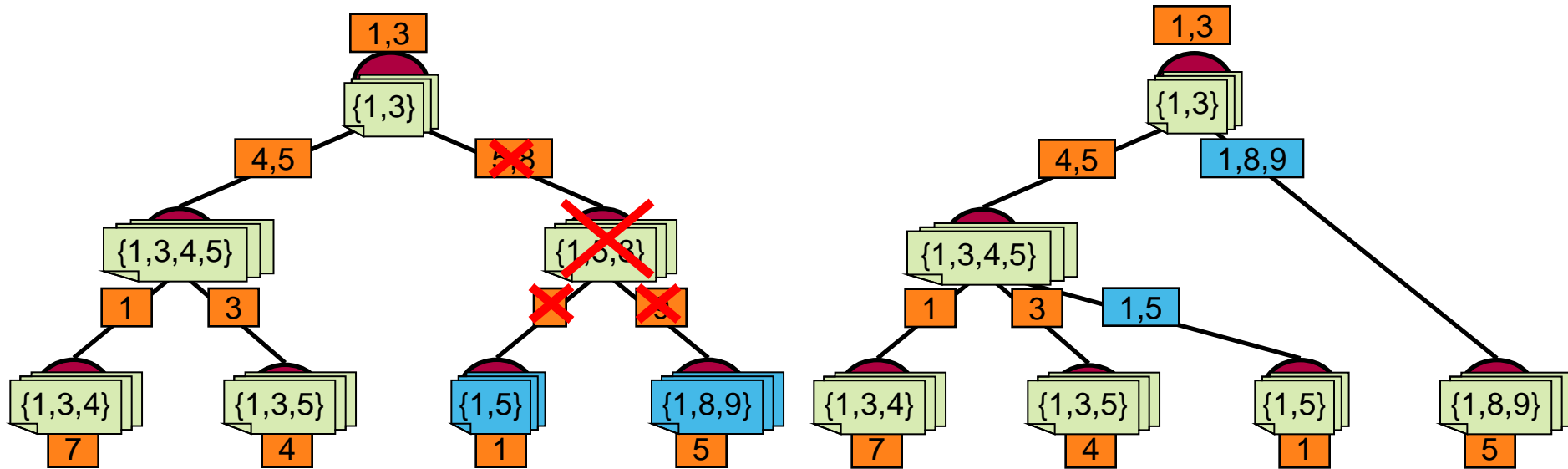
- ▶ Compensate for lost state using inherently redundant information from surviving processes
 - Avoid overhead of explicit data replication
- ▶ **State composition**
 - Lightweight mechanism for idempotent aggregations
- ▶ **State decomposition**
 - For non-idempotent aggregations
 - Requires two coordination phases
 - No overhead in the absence of failures
 - $O(\log(N))$ processes participate in failure recovery

State Composition Example

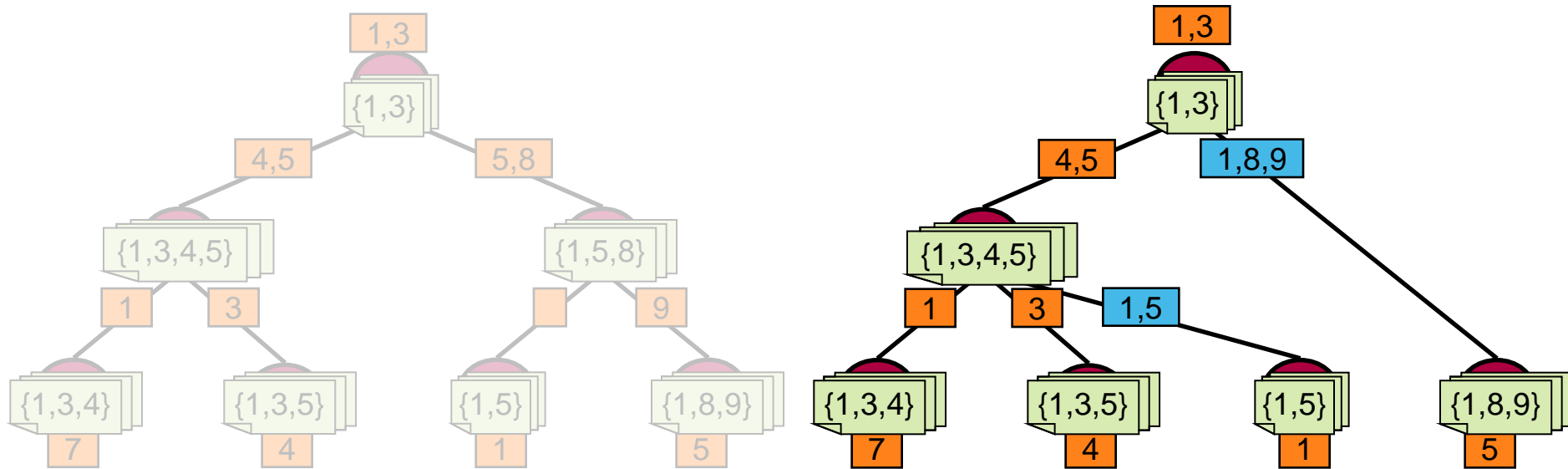


Use orphans states
to compensate for failure

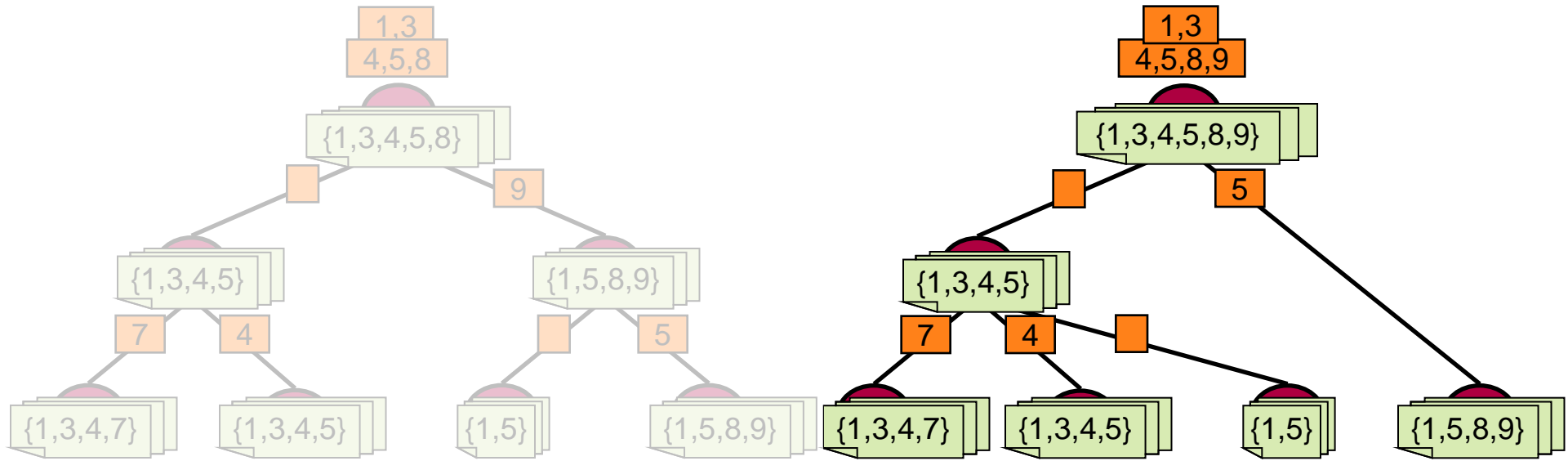
State Composition Example



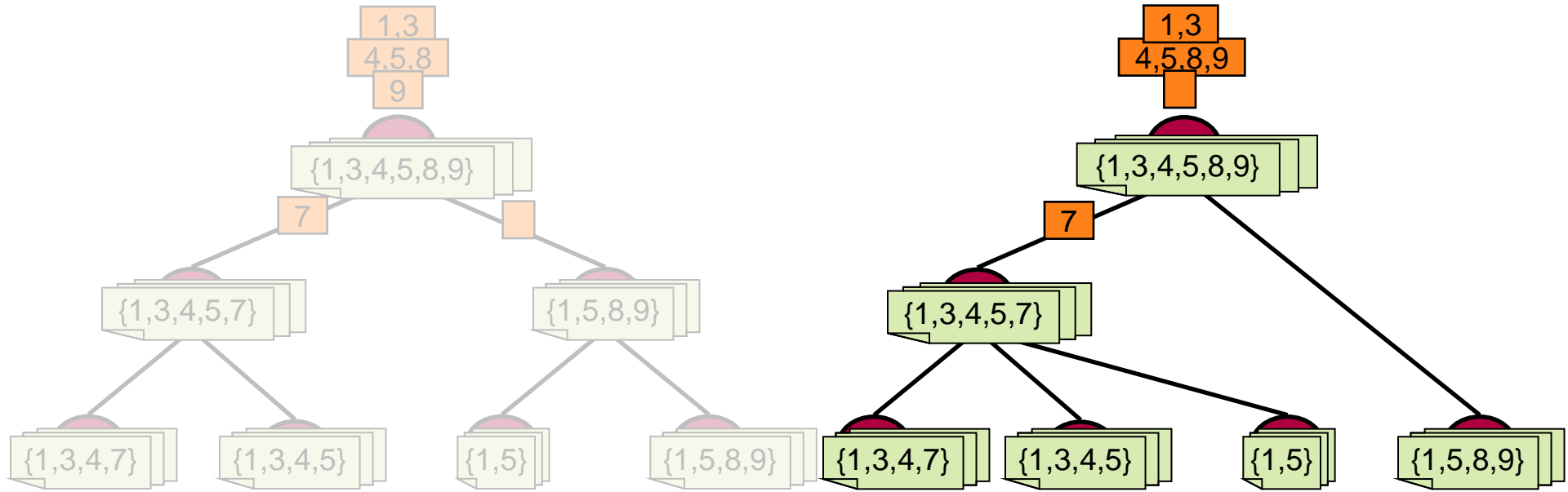
State Composition Example



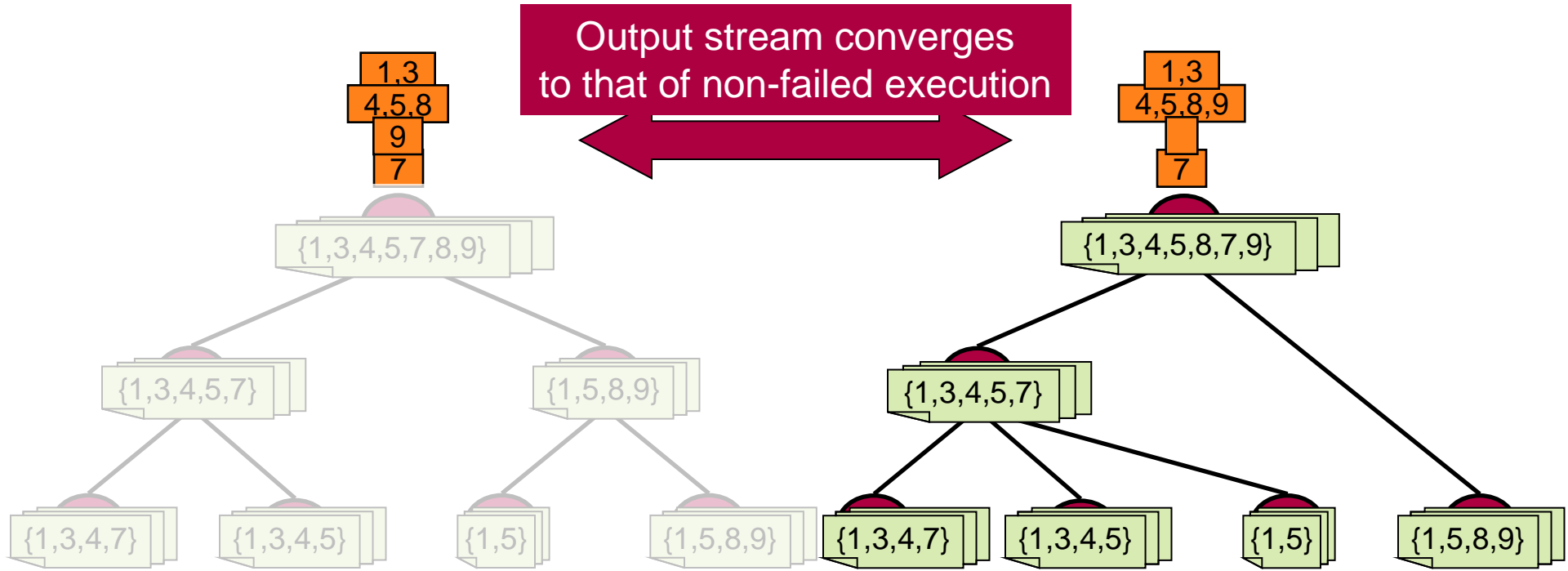
State Composition Example



State Composition Example

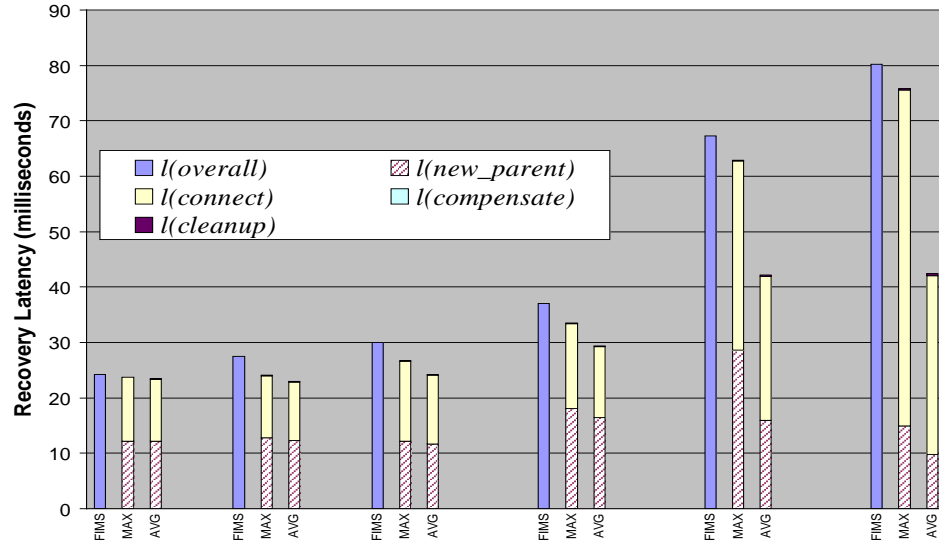


State Composition Example

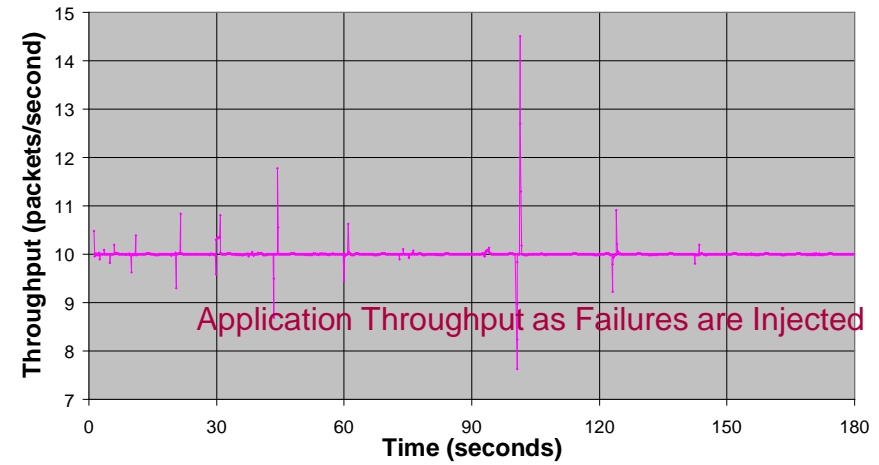


Robust, Scalable Data Aggregation

Failure Recovery Latency



Fan-out at Failed Process



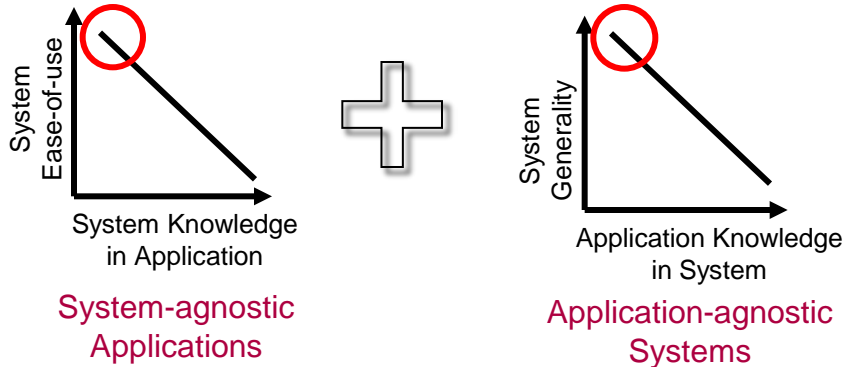
Application Throughput as Failures are Injected

Autonomous Middleware

Autonomous Middleware

Goal

Efficient, scalable systems from:



Approach: Dynamic, Autonomous Operation

Self-configuring: Automatic TBÖN topology configuration

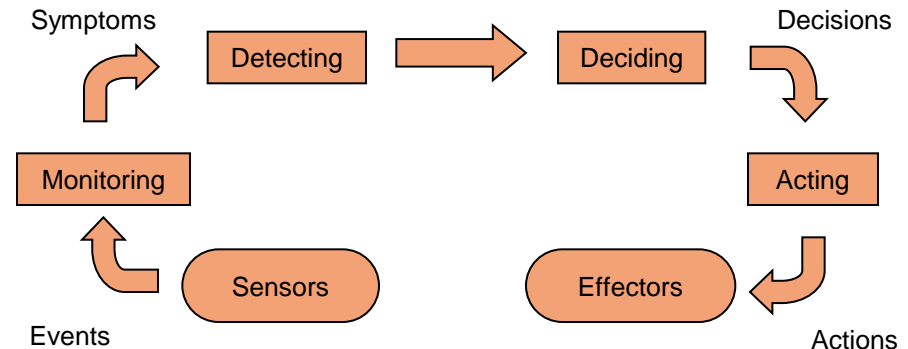
Self-monitoring: TBÖN health and performance

Self-healing: TBÖN Fault tolerance and failure recovery

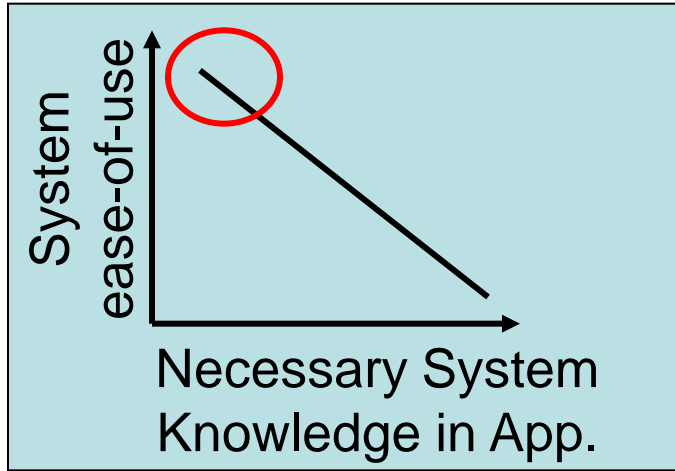
Self-optimizing: Dynamic TBÖN reconfiguration to improve performance

Challenges

- Reliable service at scale
- Choosing the “best” TBÖN topologies?
 - Load and system characteristics may vary over time
- Online improvement of TBÖN performance?
 - Throughput, latency, resource consumption, startup costs, ...
- Flexible, elegant solution space

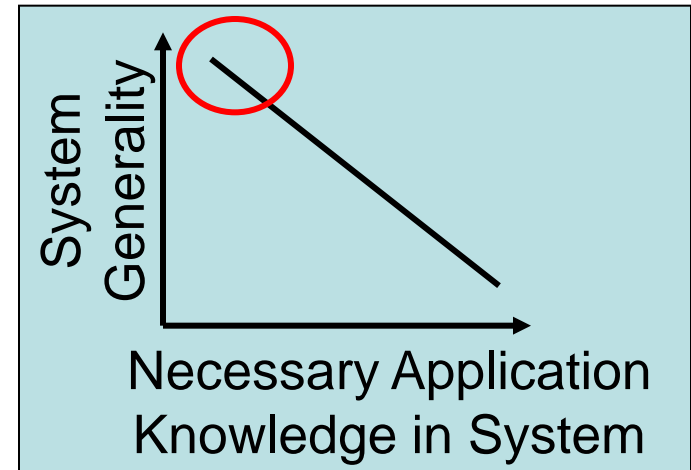


Problem Statement: Application Efficiency and Scalability



1. How much system-specific knowledge does application (developer) need?

2. How much application-specific knowledge does system (developer) need?



3. How far can we get answering "NONE" and "NONE"? ²⁷

The Approach: An Autonomous TB• N Infrastructure

TB• N Autonomy aka the self-* properties:

- Self-configuring
 - Automatic TB• N topology configuration
- Self-monitoring
 - TB• N health and performance
- Self-healing
 - TB• N Fault tolerance and failure recovery
- Self-optimizing
 - Dynamic TB• N reconfiguration to improve performance

Research Challenges

- How can we provide a reliable TB• N service in the presence of failures?
- How do we choose the “best” TB• N topologies?
 - Application load and system characteristics may vary over time
- How can we dynamically improve TB• N performance?
 - Throughput, latency, resource consumption, startup costs, ...
- Can we design a flexible, elegant solution space?

“Performance Failures”

- What is a performance failure?
 - Generally, a sub-optimal topology, or
 - Realizing (much) less than optimal performance
 - Data aggregation latency and throughput
 - Resource under-utilization
 - Imbalanced topologies
 - Per application?
 - Per flow/stream?

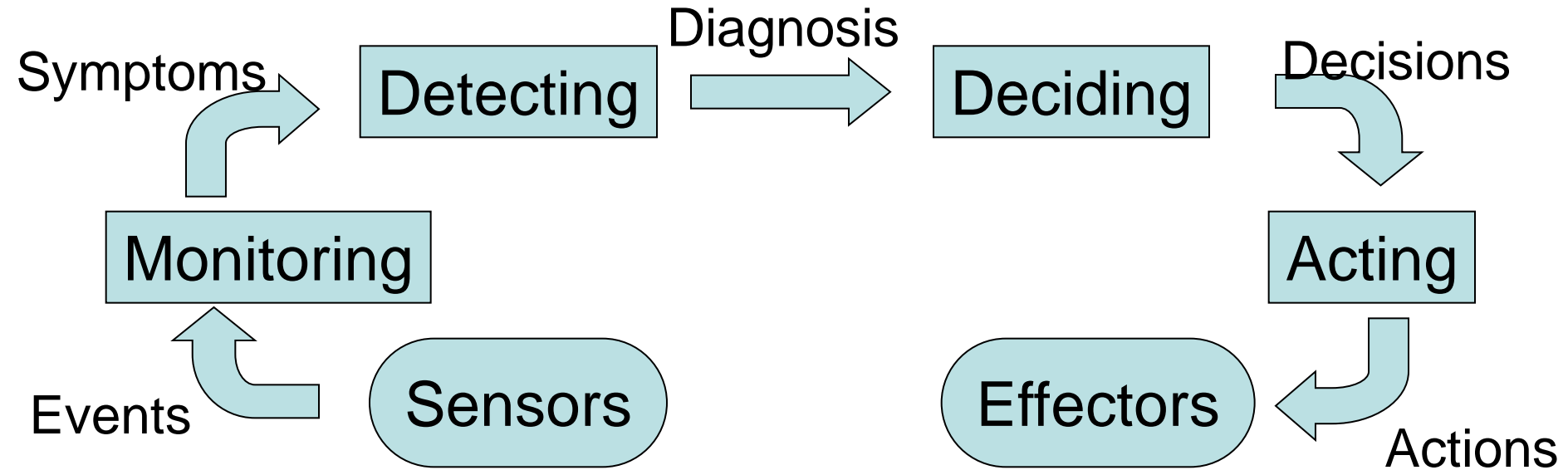
Per Flow Topologies

- “best” topology depends upon
 - Participating end-points
 - Data aggregation operation
 - Application data rate
 - ...
- “best” is different for different streams!
 - How can we efficiently enable different topologies for different flows?

Thanks to Jack and all the ICLers!

Bonus Material

TB• N Components for Autonomy



Key Challenges:

- Decentralization
- Low (background) overhead
- Rapid execution
- Must provide more benefits than drawbacks!

Example TBON Reductions

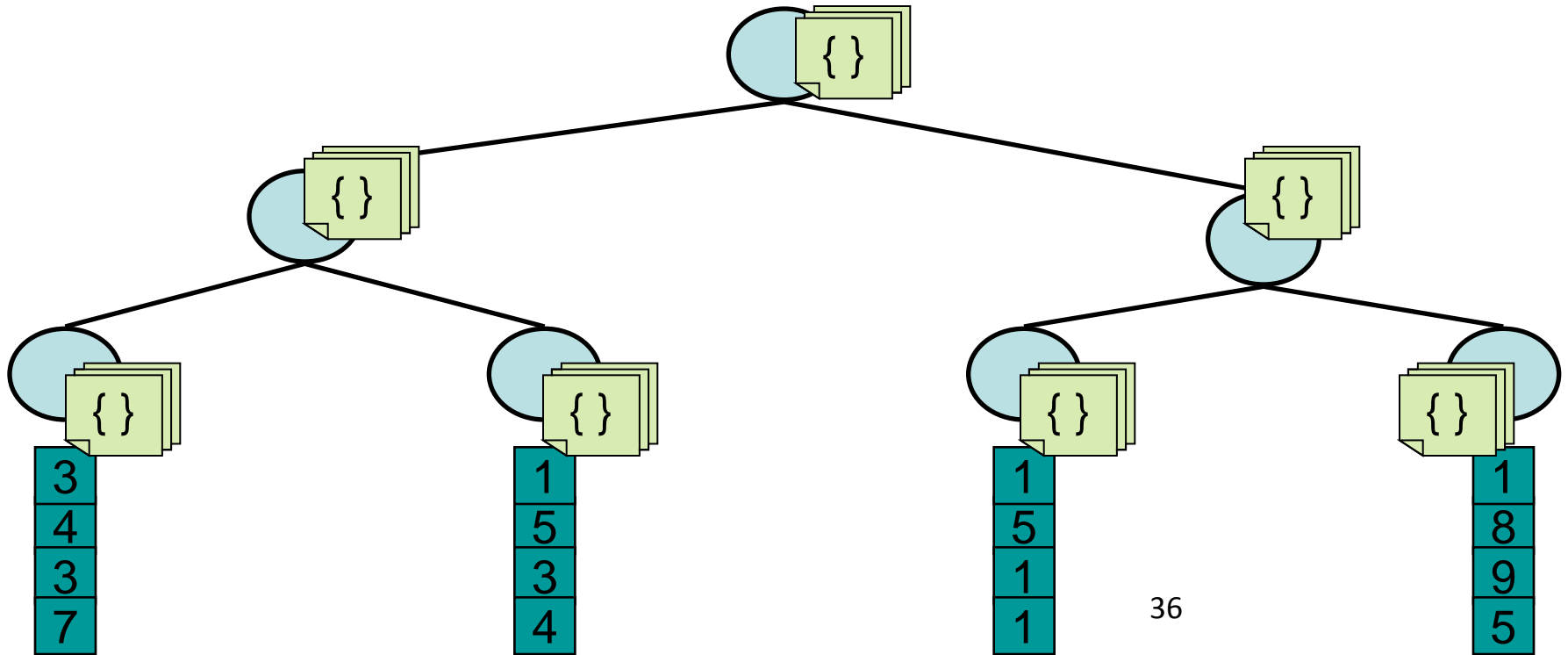
▶ Simple

- Min, max, sum, count, average
- Concatenate

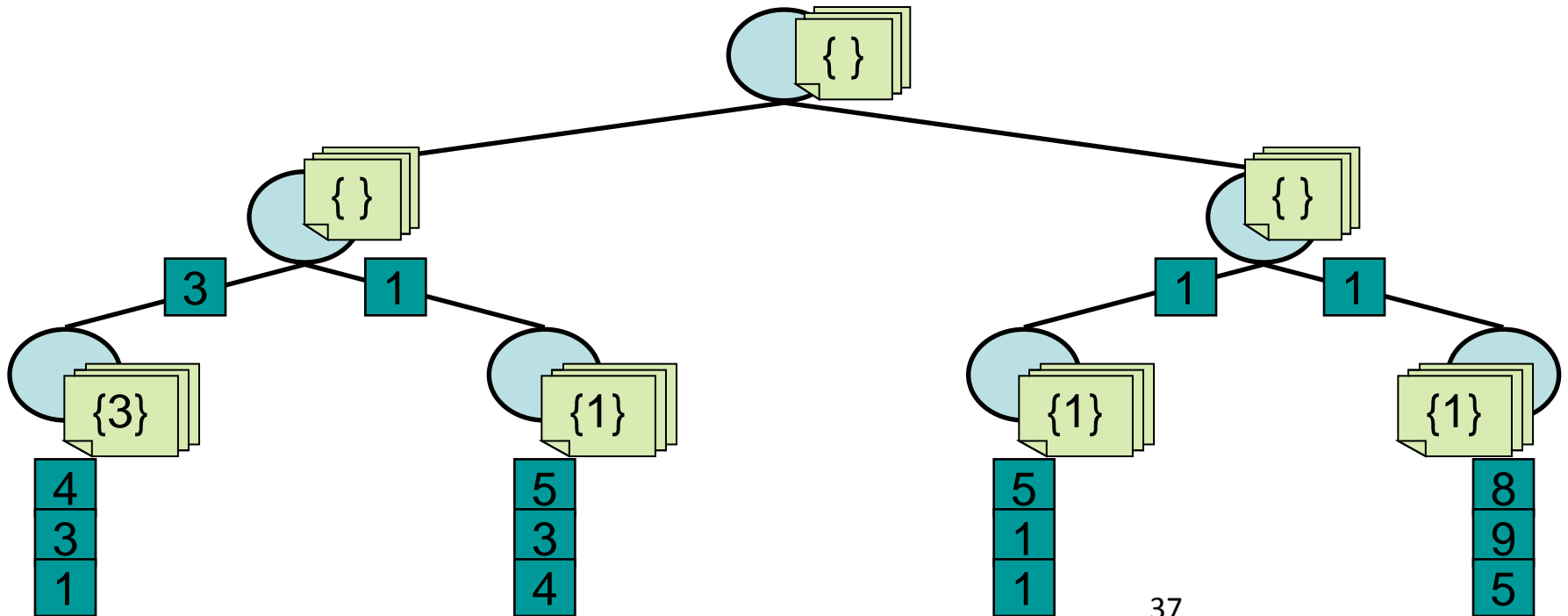
▶ Complex

- Clock synchronization [Roth, Arnold, Miller '03]
- Time-aligned aggregation [Roth, Arnold, Miller '03]
- Vision algorithms [Arnold, Pack, Miller '06]
- Graph Analysis [Arnold et al. '07], [Roth, Miller '05]
- Equivalence relations

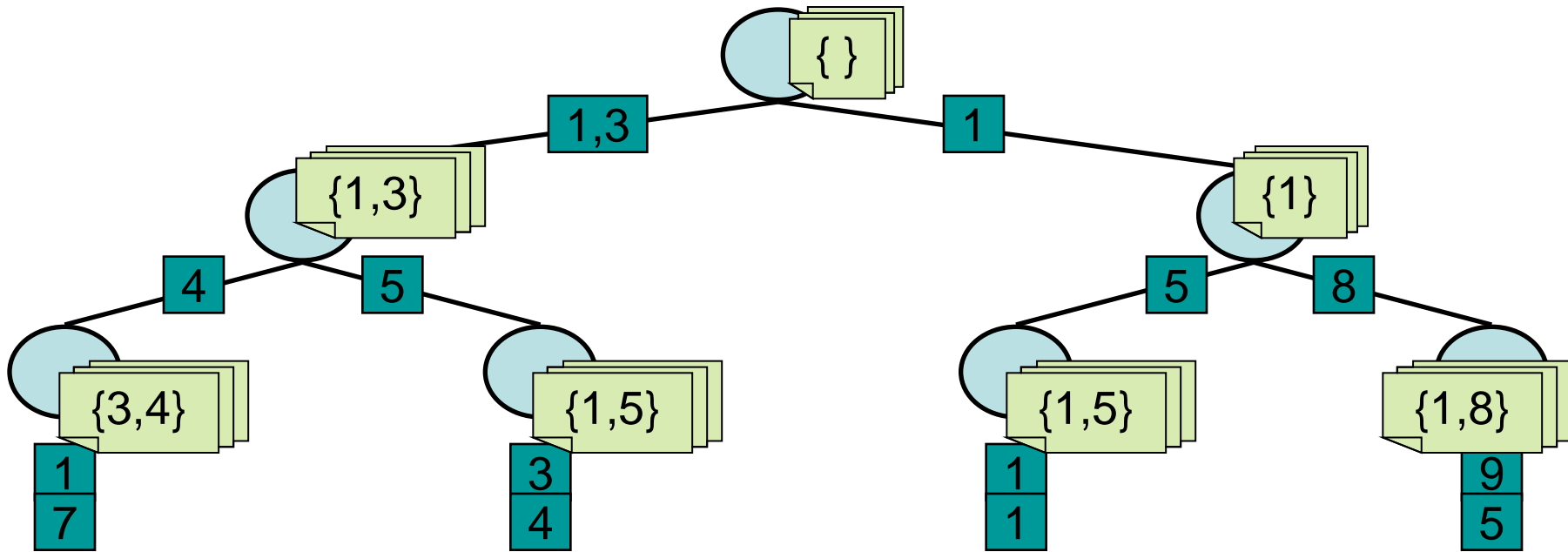
Integer Union Example



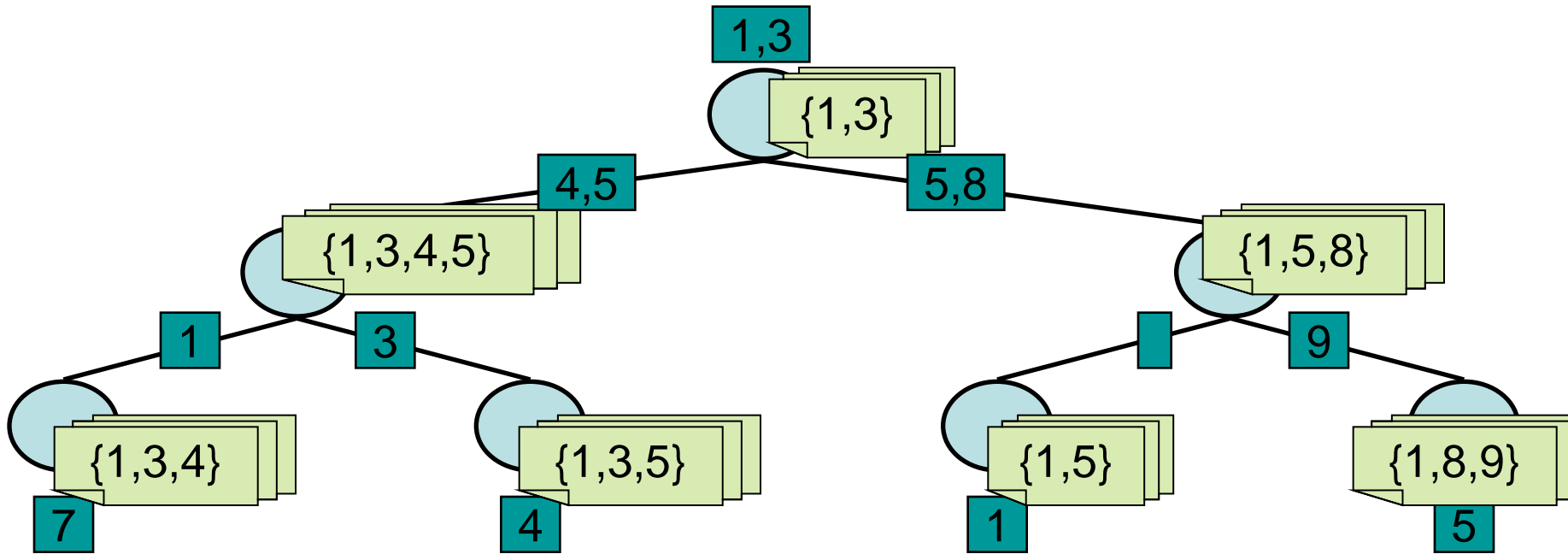
Integer Union Example



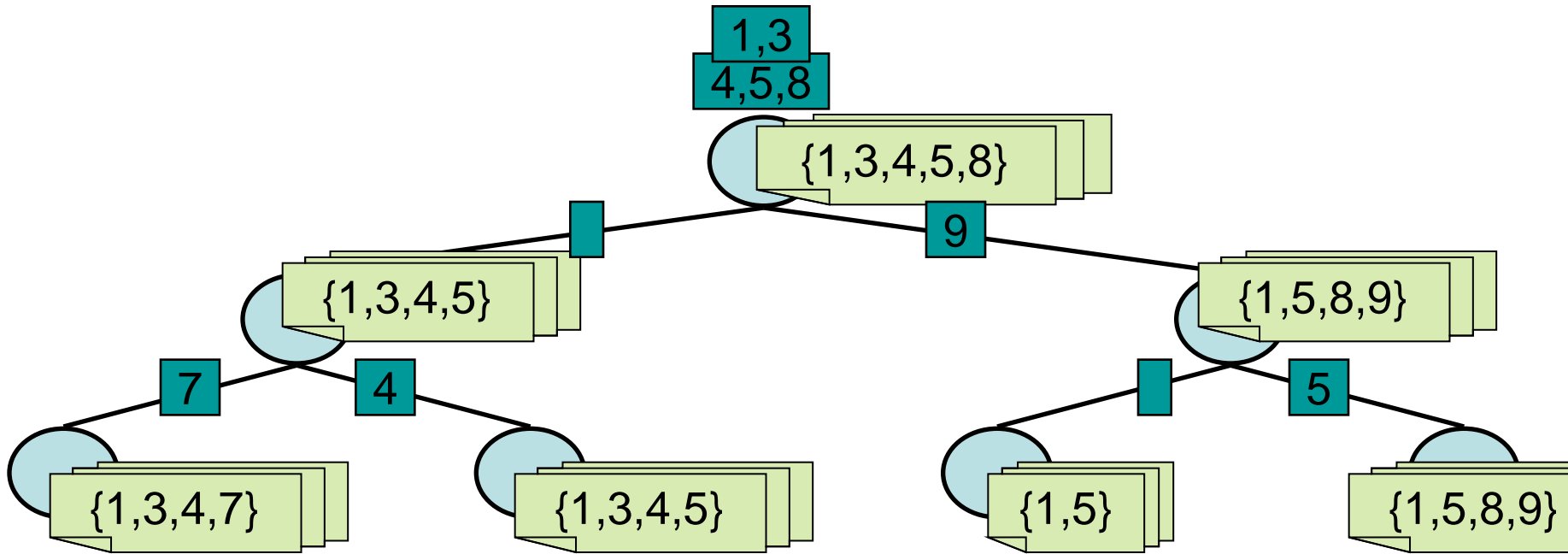
Integer Union Example



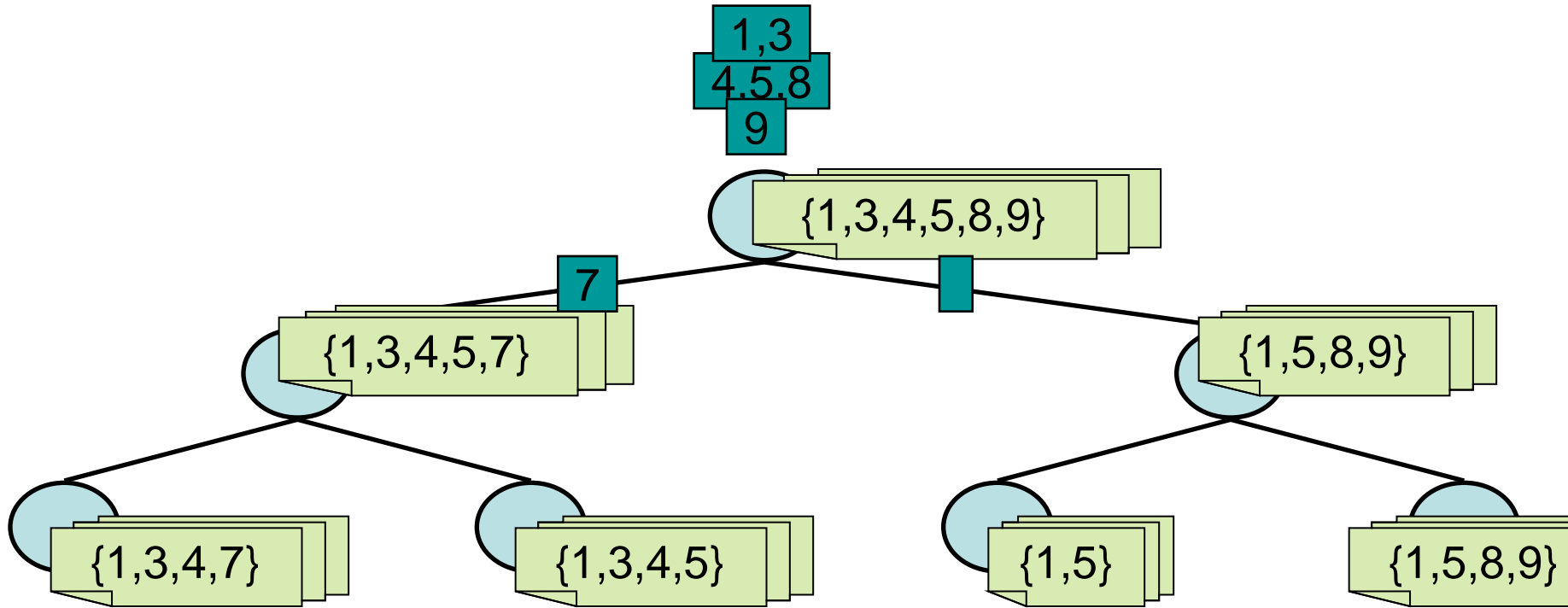
Integer Union Example



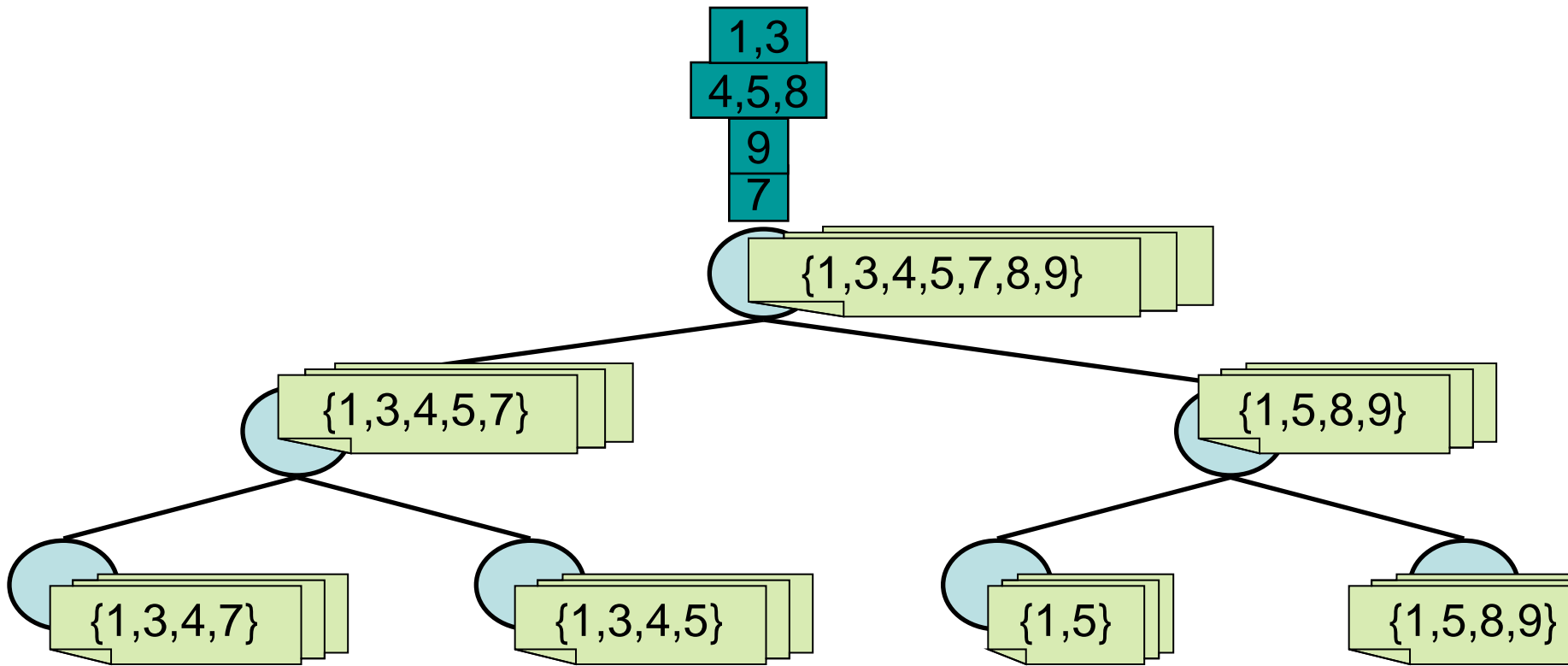
Integer Union Example



Integer Union Example



Integer Union Example



MRNet Front-end Interface

```
front_end_main(){
    Network * net = new Network (topology_file);

    Communicator * comm = net->
        get_BroadcastCommunicator();

    Stream * stream =
        new Stream( comm, IMAX_FILTER, WAITFORALL);

    stream->send( "%s", "go" );

    stream->recv( "%d", &result);
}
```

MRNet Back-end Interface

```
back_end_main(){
    Stream * stream;
    Packet *p;
    char * s;

    Network * net = new Network();

    net->recv(&tag, &p, &stream);
    p->unpack( "%s", &s );

    if(s == "go"){
        stream->send("%d", rand_int);
    }
}
```

MRNet Filter Interface

```
imax_filter(vector<Packet> packets_in,  
            vector<Packet> packets_out)  
{  
    for( i=0; i<packets_in.size; i++){  
        result = max( result,  
                     packets[i].get_int());  
    }  
  
    Packet p("%d", result);  
  
    packets_out.pushback(p);  
}
```